An Investigation of Multi-objective Genetic Algorithms for Encrypted Traffic Identification

Carlos Bacquet, A. Nur Zincir-Heywood, and Malcolm I. Heywood

Dalhousie University, Faculty of Computer Science
{bacquet,zincir,mheywood}@cs.dal.ca

Abstract. The increasing use of encrypted traffic combined with non-standard port associations makes the task of traffic identification increasingly difficult. This work adopts a multi-objective clustering approach to the problem in which a Genetic Algorithm performs both feature selection and cluster count optimization under a flow based representation. Solutions do not use port numbers, IP address or payload. Performance of the resulting model provides 90% detection 0.8% false positive rates with 13 clusters supported by 14 of the original 38 features.

1 Introduction

An important part of network management requires the accurate identification and classification of network traffic [3]. Network administrators are normally interested in identifying application types for decisions regarding both bandwidth management and quality of service [3]. A particularly interesting area in network traffic identification pertains to encrypted traffic, where the fact that the payload is encrypted represents an additional degree of uncertainty. Specifically, many traditional approaches to traffic classification rely on payload inspection, which become unfeasible under packet encryption [8, 10, 5, 4, 9]. An alternative to payload inspection would be the use of port numbers to identify application types. However, this practice has become increasingly inaccurate, as users are now able to arbitrarily change the port number to deceive security mechanisms [3, 10, 5, 9, 4]. In short, the traditional approaches are unable to deal with the identification of encrypted traffic.

In this work our objective is the identification of encrypted traffic, where Secure Shell (SSH) is chosen as an example encrypted application. While SSH is typically used to remotely access a computer, it can also be utilized for "tunneling, file transfers and forwarding arbitrary TCP ports over a secure channel between a local and a remote computer" [3]. These properties of SSH make it an interesting encrypted application to focus on. From the traffic identification perspective a Multi-Objective Genetic Algorithm (MOGA) is pursued; where this facilitates the dual identification of appropriate (flow) feature subspace and clustering of traffic types. Such a system assumes that the resulting clusters partition traffic into encrypted/not encrypted. Performance of the system achieves detection rates above 90% and false positive rates below 1%.

Á. Herrero et al. (Eds.): Computational Intel. in Security for Info. Systems, AISC 63, pp. 93–100.
springerlink.com

2 Previous Work

Given the limitations of port number analysis and payload inspection, several previous attempts to identify encrypted traffic have worked with the statistics of the flow [3, 10, 5]. A number of these attempts have employed supervised learning methods. However, these classifiers have uncertain generalization properties when faced with new data [10, 5]. One alternative to classifiers is the use of clustering mechanisms or unsupervised learning methods. The following is an analysis of previous work with unsupervised methods. To the best of our knowledge, this is the first work that utilizes a genetic algorithm for the dual problem of feature selection and clustering for encrypted traffic identification.

Recently, Siqueira *et al.* presented a clustering approach to identify Peer-to-Peer (P2P) versus non-P2P in TCP traffic [8]. A total of 249 features from the headers of the packets were considered, out of which the best 5 were selected to parameterize clusters. The selection of the features was based on the variance of their values, such that the higher the variance, the better the perceived discrimination of a feature. This methodology enabled the authors to achieve results of 86.12% detection rate for P2P applications with an average accuracy of 96.79%. However, the method utilized the port number as one of the five features. Erman *et al.* also presented a clustering approach for the network traffic identification problem [5]. They evaluated three clustering algorithms: K-means, DBSCAN and AutoClass. The authors used two datasets, one of them (Auckland IV) containing DNS, FTP, HTTP, IRC, LimeWire, NNTP, POP3, and SOCKS; and the other (Calgary trace), containing HTTP, P2P, SMTP, and POP3. The authors found that with K-means the overall accuracy steadily improved as the number of clusters was increased. This continued until K was around 100 with the overall accuracy being 79% and 84% on each data set respectively. Thus, from their results they observed that K-means seemed to provide the best mix of properties. Bernaille *et al.* used information from the first five packets of a TCP connection to identify applications [4]. In particular, they analyzed the size of the first few packets, which captures the application's negotiation phase. They then used a K-means algorithm to cluster these features during the learning phase. After several experiments they concluded that the best K number of clusters was 50, achieving results of up to 96.9% accuracy with SSH. The authors did mention that the method is sensitive to packet order, thus a potential limitation. Yingqiu *et al.* also presented a flow based clustering approach, using K-means to build the clusters with features previously identified as the best discriminators [10]. To this end, several feature selection and search techniques are performed. After clustering, the authors applied a log transformation, which enabled them to reach an overall accuracy level of up to 90% when utilizing $K = 80$ clusters. The authors concluded this was a very promising approach, however, they also mention that they only worked with TCP. It should be noted here that none of the above work reported their false positive rates.

3 Methodology

In this section we first characterize the data utilized for our training and testing. The entire process employed in this work is outlined in Figs. 1 and 2. The data set used

was captured by the Dalhousie University Computing and Information Services Centre (UCIS) in January 2007 on the campus network between the university and the commercial Internet. Given the privacy related issues the university may face, the data was filtered to scramble the IP addresses and each packet was further truncated to the end of the IP header so that all the payload was excluded. Furthermore, the checksums were set to zero since they could conceivably leak information from short packets. However, any length information in the packet was left intact. Dalhousie traces were labeled by a commercial classification tool called PacketShaper, i.e., a deep packet analyzer [14]. PacketShaper uses Layer 7 filters (L7) to classify the applications. Given that the handshake part of SSH protocol is not encrypted, we can confidently assume that the labeling of the data set is 100% correct and provides the ground truth for our data set. We emphasize that our work did not consider any information from the handshake phase nor any part of the payload, IP addresses or port numbers. The handshake phase was used solely to obtain the ground truth to which we compare our obtained results. Also, we focus on SSH as a case study, we could have employed any other encrypted traffic protocol. However, the fact the SSH's handshake is not encrypted, allowed us to compare our obtained results with those obtained through payload inspection. In order to build training data and test data, we divided the trace into two partitions. We sampled our training data from the first partition and the test data from the remaining partition. The training data consisted of 12240 flows, including SSH, MSN, HTTP, FTP, and DNS. The test data, on the other hand, consisted of 1000 flows of each of those applications, plus 1000 flows that belonged to any of the following applications: RMCP, Oracle SQL*NET, NPP, POP3, NETBIOS Name Service, IMAP, SNMP, LDAP, NCP, RTSP, IMAPS and POP3S.

Flows are defined by sequences of packets that present the same values for source IP address, destination IP address, source port, destination port and type of protocol [8]. Each flow is described by a set of statistical features and associated feature values. A feature is a descriptive statistic that can be calculated from one or more packets. We used the NetMate [11] tool set to process data sets, generate flows, and compute feature values. In total, 38 features were obtained. Flows are bidirectional with the first packet determining the forward direction. Since flows are of limited duration, in this work UDP flows are terminated by a flow timeout, and TCP flows are terminated upon proper connection teardown or by a flow timeout, whichever occurs first. Given that there is not a general and accepted value for the flow time out, a 600 second flow timeout value was employed here; where this corresponds to the IETF Realtime Traffic Flow Measurement working groups architecture [13]. It is important to mention that only UDP and TCP flows are considered. Specifically, flows that have no less than one packet in each direction, and transport no less than one byte of payload.

As for feature selection we have employed the model proposed by YeongSeog *et al*. [6] and modified its evolutionary component to follow the model proposed by Kumar *et al*. [7]. The latter ensures a convergence towards the Pareto-front (set of non-dominated solutions) without any complex sharing/niching mechanism. One specific property of this Genetic Algorithm (GA) is the utility of a steady-state GA, thus only one or two members of the population are replaced at a time, Fig. 2. A GA starts with a population of individuals (potential solutions to a problem), and incrementally evolves that population into better individuals, as established by the fitness criteria. Fitness is naturally relative to the population. Then, for several iterations, individuals

are selected to be combined (crossover) to create new individuals (offspring) under a fitness proportional selection operator. In order to model the problem of feature selection to the GA, each individual in the population represents a subset of features *f* and a number of clusters *K*. Specifically, an individual is a 60 bit binary string, with the first 38 bits representing the features to include and the remaining 22 bits representing the *K* number of clusters. Bits of the individuals in the initial population are initialized with a uniform probability distribution. For the feature selection, a one implies to include the feature at that index, and a zero means to discard it. The K number of clusters, on the other hand, is represented by the number of "ones" (as opposed to "zeros") contained between the 39^{th} bit and the 60^{th} bit, plus two. The reason for adding two is that zero or one cluster would never be a solution. Clusters are identified using the standard K-means algorithm, using that subset of features *f*, and the number of clusters *K,* as the input for the K-means algorithm. For this step we used the K-means algorithm provided by Weka [12]. The fitness of the individual will then depend on how well the resulting clusters perform in relation to the following four predefined clustering objectives:

- *Fwithin* (measures cluster cohesiveness, the more cohesive the better). We calculate the average standard deviation per cluster. That is, per each *i*'th cluster, the sum of the standard deviations per feature over the total number of employed features. Then *Fwithin* will be *K* over the sum of all the clusters' average standards.

- *Fbetween* (measures how separate the clusters are from each other, the more separated the better). For each pair of cluster *i* and *j* we calculate its average standard deviations and we also calculate the euclidean distance between its centroids. Then, *Fbetween* for clustetr *i* and *j* is:

$$Fbetween\_i\text{-}j = \frac{Euclidean\ distance\ from\ i\ to\ j}{\sqrt{((AveStdDev\_i)\text{^}2+(AveStdDev\_j)\text{^}2)}} \quad (1)$$

Thus, Fbetween will be the sum of all pairs of cluster's *Fbetween_i-j,* over *K.*

- *Fclusters* (measures the number of clusters *K*, "Other things being equal, fewer clusters make the model more understandable and avoid possible over fitting"[6])

$$Fclusters = 1 - \frac{K - Kmin}{Kmax - Kmin} \quad (2)$$

Kmax and *Kmin* are the maximum and minimum number of clusters.

- *Fcomplexity* (measures the amount of features used to cluster the data, this objective aims at minimizing the number of selected features)

$$Fcomplexity = 1 - \frac{d\text{-}1}{D - 1} \quad (3)$$

D is the dimensionality of the whole dataset and *d* is the number of employed features.

Instead of combining these objectives into a single objective, this model followed a multi-objective approach, which has the goal to approximate to the Pareto front, or set of non-dominated solutions. Informally, a solution is said to dominate another if it has higher values in at least one of the objective functions (*Fwithin*, *Fbetween*, *Fclusters*, *Fcomplexity*), and is at least as good in all the others. After the objective values for each individual have been assessed, individuals are assigned with ranks, which indicate how many individuals dominate that particular individual. Finally, the fitness of the individuals is inversely proportional to their ranks, which is used to build a roulette wheel that is ultimately used for parental selection under the aforementioned steady state model. The initial population is evolved for 5000 epochs, after which we consider the set of non-dominated individuals from it (individuals whose ranks equal to 1). These individuals correspond to the set of potential solutions. The evolutionary component of the algorithm is then terminated and the best individual in the set of non-dominated solutions (the one that better identifies SSH traffic) is identified. We take each individual from the set of non dominate solutions and label its clusters as either SSH or non-SSH, based on the labels in the training data. If the majority of the flows in a cluster have SSH labels, then that cluster is labeled as SSH, otherwise it is labeled as non-SSH. Then, the post-training phase consists of testing each individual in our labeled training data (used to build the clusters on), to identify the solution with best classification rate, which will be the final solution.

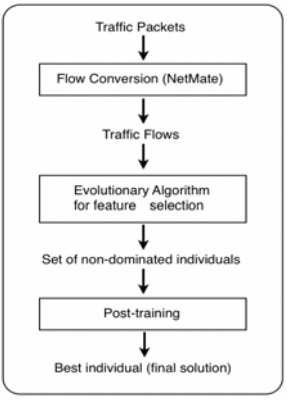

Fig. 1. System Diagram

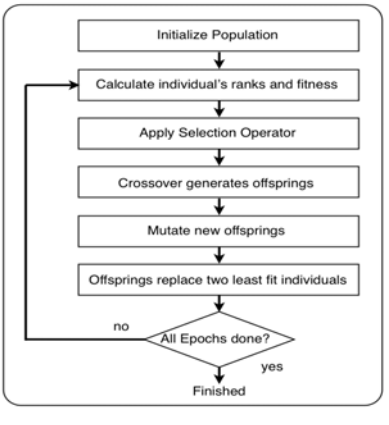

Fig. 2. Evolutionary Component Diagram

4 Results

In traffic classification, two metrics are typically used to quantify performance: Detection Rate (DR), Eq. 4, and False Positive Rate (FPR), Eq 5. In this case, detection rate will reflect the number of SSH flows correctly classified, and FPR will reflect the number of Non-SSH flows incorrectly classified as SSH. As we can observe, a high DR and a low FPR would be the desired outcomes.

$$DR = 1 - \frac{\#\,false\_negatives}{total\_number\_of\_SSH\_flows} \quad (4)$$

$$FPR = \frac{\#\,false\_positives}{total\_number\_of\_non\_SSH\_flows} \quad (5)$$

false_negatives mean SSH traffic incorrectly classified as non-SSH traffic. In these experiments we compared our proposed system, MOGA, with K-means algorithm using the same feature set and the same data sets. Weka's K-means clusters were built with values of *K* from 5 to 100. The results show more than 94% DR with very high FPR (more than 3.4%). Table 1.

Table 1. Baseline – K-means on Dalhousie data sets

K	DR	FPR	K	DR	FPR	K	DR	FPR	K	DR	FPR
5	0.94	0.12	25	0.95	0.04	45	0.97	0.10	80	0.98	0.12
10	0.97	0.16	30	0.97	0.16	50	0.97	0.15	90	0.97	0.06
15	0.97	0.18	35	0.97	0.16	60	0.97	0.07	100	0.97	0.07
20	0.95	0.04	40	0.96	0.034	70	0.98	0.14			

On the other hand, the MOGA was run 25 times, producing a combined of 783 non-dominated individuals. Out of those individuals, we identified in the post-training phase the one that had the lowest FPR and a DR close to 90%. We considered that individual to be our final solution. Fig. 3 displays the plot of the candidate solutions in the post-training phase. The *x*-axis represents the FPR and the *y*-axis represents the DR. The selected individual, which is represented by a larger black square instead of a gray diamond, achieved a detection rate of 90% and a false positive rate of 0.08% in the post training phase. That same individual achieved a detection rate of 92% and a false positive rate of 0.8% in the test data. This final solution employed only 14 out of the 38 available features, and it clustered the data into 13 clusters. Thus, this solution not only achieved very promising results in terms of detection rate and false positives rate, but it also considerably decreased the number of employed features, and clustered the data with a relatively low value of *K* clusters, both very desirable outcomes. The features employed by our final solution are displayed in Table 2. In addition, an analysis of the best 10 individuals reveals that these other well performing individuals also employed some of these features. Table 3 displays which features were the most recurrent among the top ten individuals.

Finally, it is also important to mention the frequency with which we were able to obtain these kinds of results in our conducted runs. Given the stochastic component of evolutionary algorithms, it is expected that some runs will be better than others. In this case, we were able to obtain individuals with detection rates above 90% and false positive rates below 1% in 21 out of the 25 conducted runs.

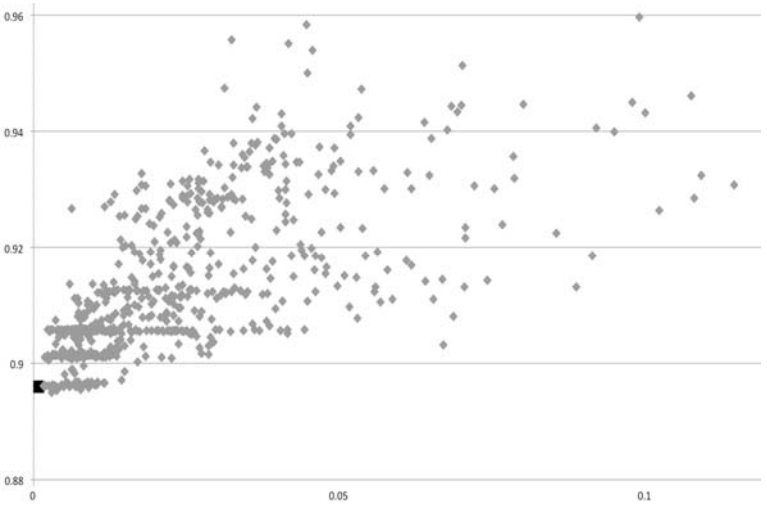

Fig. 3. Plot of Detection rate versus false positives in post-training phase

Table 2. Features employed by the best individual

Feature	Feature name
Total back packets (total_bpackets)	Min back inter arrival time (min_biat)
Total back volume (total_bvolume)	Mean back inter arrival time (mean_biat)
Mean fwrd packet length (mean_fpktl)	Max back inter arrival time (max_biat)
Max fwrd packet length (max_fpktl)	Sub-flow fwrd packets (sflow_fpackets)
Std fwrd packet length (std_fpktl)	Sub-flow fwrd bytes (sflow_fbytes)
Min back packet length (min_bpktl)	Back push counters (bpsh_cnt)
Std fwrd inter arrival time (std_fiat)	Fwrd urgent counters (furg_cnt)

Table 3. Most recurrent features employed by top ten individuals

Feature	#	Feature	#
Min back inter arrival time (min_biat)	10	Max back packet length (max_bpktl)	5
Total fwrd packets (total_fpackets)	7	Min fwrd inter arrival time (min_fiat)	5
Total back packets (total_bpackets)	7	Std fwrd inter arrival time (std_fiat)	5
Std back inter arrival time (std_biat)	7	Total back volume (total_bvolume)	5
Fwrd urgent counters (furg_cnt)	7	Sub-flow fwrd packets (sflow_fpackets)	5
Mean fwrd packet length (mean_fpktl)	6	Sub-flow back packets (sflow_bpackets)	5
Std fwrd packet length (std_fpktl)	5	Back urgent counters (burg_cnt)	5

5 Conclusions

A steady state MOGA is applied to the dual problem of feature subspace selection and clustering of encrypted traffic. Compared to basic K-means algorithm, the proposed method not only considerably decreases the number of features needed for clustering the flows, but it also made it possible to cluster the data with a very small value of *K*.

Both of which are very desirable advantages for a potential implementation of an encrypted traffic detection system. The results were presented in terms of DR and FPR. Our best individual achieved a detection rate of 92% and a false positive rate of 0.8%, whereas it employed only 14 out of the 38 available features and clustered the data in only 13 clusters. For future work, we aim to apply this methodology to other types of encrypted traffic such as Skype.

References

[1] Alshammari, R., Zincir-Heywood, N.: Generalization of Signatures for SSH Traffic Identification. In: IEEE Symposium Series on Computational Intelligence (2009)
[2] Alshammari, R., Zincir-Heywood, N.: Investigating two different approaches for encrypted traffic classification. In: Sixth Annual Conference on Privacy, Security and Trust, pp. 156–166 (2008)
[3] Alshammari, R., Zincir-Heywood, N.: A flow based approach for SSH traffic detection. ISIC. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 296–301 (2007)
[4] Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., Salamatian, K.: Traffic classification on the fly. SIGCOMM Comput. Commun. Rev. 36(2), 23–26 (2006)
[5] Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: SIGCOMM Workshop on Mining Network Data, pp. 281–286. ACM, New York (2006)
[6] YeongSeog, K., Street, W.N., Menczer, F.: Feature selection in unsupervised learning via evolutionary search. In: Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 365–369. ACM, New York (2000)
[7] Kumar, R., Rockett, P.: Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution: A pareto converging genetic algorithm. Evol. Comput. 10(3), 283–314 (2002)
[8] Siqueira Junior, G.P., Bessa Maia, J.E., Holanda, R., Neuman de Sousa, J.: P2P traffic identification using cluster analysis. In: First International Global Information Infrastructure Symposium (GIIS), pp. 128–133 (2007)
[9] Wright, C., Monrose, F., Masson, G.: On inferring application protocol behaviors in encrypted network traffic. J. Mach. Learn. Res. 7, 2745–2769 (2006)
[10] Yingqiu, L., Wei, L., Yun-Chun, L.: Network traffic classification using K-means clustering. In: Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS), pp. 360–365. IEEE Computer Society, Washington (2007)
[11] NetMate, `http://www.ip-measurement.org/tools/netmate/`
[12] WEKA Software, `http://www.cs.waikato.ac.nz/ml/weka/`
[13] IETF, `http://www3.ietf.org/proceedings/97apr/97apr-final/xrtftr70.htm`
[14] PacketShaper, `http://www.packeteer.com/products/packetshaper`